Revisiting Metric Learning for SPD Matrix based Visual Representation

Luping Zhou, Lei Wang
University of Wollongong, Australia
lupingz,leiw@uow.edu.au

Jianjia Zhang
CSIRO, Australia
seuzjj@gmail.com

Yinghuan Shi, Yang Gao
Nanjing University, China
syh,gaoy@nju.edu.cn

Abstract

The success of many visual recognition tasks largely depends on a good similarity measure, and distance metric learning plays an important role in this regard. Meanwhile, Symmetric Positive Definite (SPD) matrix is receiving increased attention for feature representation in multiple computer vision applications. However, distance metric learning on SPD matrices has not been sufficiently researched. A few existing works approached this by learning either $d^2 \times p$ or $d \times k$ transformation matrix for $d \times d$ SPD matrices. Different from these methods, this paper proposes a new member to the family of distance metric learning for SPD matrices. It learns only d parameters to adjust the eigenvalues of the SPD matrices through an efficient optimisation scheme. Also, it is shown that the proposed method can be interpreted as learning a sample-specific transformation matrix, instead of the fixed transformation matrix learned for all the samples in the existing works. The optimised d parameters can be used to "massage" the SPD matrices for better discrimination while still keeping them in the original space. From this perspective, the proposed method complements, rather than competes with, the existing linear-transformation-based methods, as the latter can always be applied to the output of the former to perform distance metric learning in further. The proposed method has been tested on multiple SPD-based visual representation data sets used in the literature, and the results demonstrate its interesting properties and attractive performance.

1. Introduction

SPD matrix based visual representation (SPD-Rep) has been used in a spectrum of visual recognition tasks, as it benefits from the advantages of considering high order statistics of the imaging signals. The most common form of SPD-Rep is covariance matrix, widely employed in applications such as texture classification [14], face recognition [13], action recognition [31, 10], pedestrian detection [16, 12], and image set classification [7, 11], etc., as either region descriptor or generic feature representation. In addition to covariance matrix, other forms of SPD-Rep have also been seen in the recent literature. For example, the work in [22] uses a SPD kernel matrix as generic visual representation to model the nonlinear relationship of features, achieving the state-of-the-art performance in some action recognition tasks. The resulting matrix of Gaussian distribution has been exploited in [24] to capture the probabilistic model of object variations for image set classification. In medical image analysis, SPD matrices have long been used for diffusion tensor data [1], and correlation matrix and inverse covariance matrix have been employed to model the interaction of brain regional imaging signals [8, 25].

This newly surging category of visual representation casts new challenges to many traditional recognition methods since SPD matrices reside on a specific Riemannian manifold instead of a flat vector space. To cater for this geometric structure, various methods have been proposed to improve the similarity comparison of SPD matrices. The typical ones include the affine-invariant Riemannian metric (AIRM) [4], Log Euclidean metric [1, 20, 23], Cholesky distance [3], Power Euclidean distance [3], Stein divergence [15], etc. On the other hand, distance metric learning is important in providing good metrics critical to the success of many visual recognition tasks, and has been an intensively researched problem [29]. The classic metric learning methods become inadequate for SPD-Rep, since they are usually defined on vector-formed data and Euclidean geometric structure. Despite its importance, the distance metric learning tailored for SPD-Rep has not been sufficiently researched. A few pioneering works have utilised the Log Euclidean metric to adapt classic distance metric learning methods to the domain of SPD matrices. For example, in [19], each $d \times d$ SPD matrix takes a logarithm operation and is then vectorised as the input to the classic ITML (Information-Theoretic Metric Learning) [2] algorithm for metric learning. This leads to the learning of a $d^2 \times d^2$ [1] Mahalanobis matrix, which quickly becomes intractable with the increase of d. As an improvement, another distance metric learning work in [9] learns a $d \times k$ ($k \leq d$) transforma-

[1] or learning a $\frac{d\times(d+1)}{2} \times \frac{d\times(d+1)}{2}$ Mahalanobis matrix considering the symmetry of SPD matrices.

tion matrix to project the logarithm of SPD matrices from the original tangent space to a new (lower-dimensional) tangent space for better discrimination.

In this paper, we propose a distance metric learning algorithm for SPD matrices from a new perspective. Different from the existing work in [19], the proposed method does not involve any vectorisation of SPD matrices. Also, it does not project these matrices to another lower-dimensional space as in [9], but performs the metric learning while maintaining the SPD matrices in its original space.

Specifically, it is well known that a SPD matrix can be decomposed into pairs of eigenvectors and eigenvalues. The former encodes the essential information on feature correlation or the subspace of data, while the latter reflects the significance of different eigen-modes in characterising the data. Given a set of $d \times d$ SPD matrices, the proposed method aims to perform metric learning by optimally adjusting the d eigenvalues for greater discrimination. Leaving the eigenvectors untouched not only preserves the related essential information on the data, but also considers the fact that some SPD-matrix-related metrics, e.g., the Log Euclidean metric, are invariant to unary transformations that map a set of eigenvectors to another set of eigenvectors.

The proposed method learns only d variables via an efficient optimisation scheme, in contrast to the potentially d^2 or even d^4 variables learned in [9, 19]. Moreover, we provide theoretical analysis of the relationship between the proposed method and the existing metric learning methods for SPD matrices. Interpreting our method from their perspective, we show that our method essentially learns a sample-specific transformation matrix, and this distinguishes itself from the existing metric learning methods that learn a fixed transformation matrix for all the samples [9, 19]. In addition, the learned d variables can be used to "massage" the data while the resulting data still stay in the original space and maintain the original physical meaning of features. This property could be useful for some applications like medical imaging analysis that cater for the interpretation of the results. Also, from this perspective, the proposed method complements, rather than competes with, the existing methods [9, 19], as the latter can be applied on the results of the former to further improve the performance. By demonstrating the performance of the proposed method on various SPD representations, we show that the proposed method could enrich the current research on distance metric learning for SPD matrices through introducing a novel and efficient member into this family.

2. Background

2.1. Log Euclidean Metric

The space of $d \times d$ SPD matrices (denoted as $\mathbb{S}_d^+$) forms a Lie group that is a Riemannian manifold [1], rather than a linear space. All derivatives at a point $\mathbf{S}$ on $\mathbb{S}_d^+$ form a tangent space $T_{\mathbf{S}}\mathbb{S}_d^+$ via the matrix logarithm map $\log_{\mathbf{S}} : \mathbb{S}_d^+ \mapsto T_{\mathbf{S}}\mathbb{S}_d^+$, where $T_{\mathbf{S}}\mathbb{S}_d^+$ is a vector space with inner product. The Log Euclidean metric framework shows that $\mathbb{S}_d^+$ admits bi-invariant metrics and the geodesic distance corresponding to the bi-invariant metrics equals the distance induced by the inner product in the tangent space $T_{\mathbf{S}}\mathbb{S}_d^+$.

Specifically, in the framework of Log Euclidean metric, the logarithmic multiplication operation $\odot : \mathbb{S}_d^+ \times \mathbb{S}_d^+ \mapsto \mathbb{S}_d^+$ is defined as $\mathbf{S}_1 \odot \mathbf{S}_2 = \exp(\log(\mathbf{S}_1) + \log(\mathbf{S}_2))$, which generalises the matrix multiplication when two SPD matrices do not commute in the matrix sense. By the commutativity of $\odot$, the $\mathbb{S}_d^+$ space is an Abelian group that admits bi-invariant metrics, i.e. metrics that are invariant by multiplication and inversion. It can be shown that any metric $\langle \cdot, \cdot \rangle$ on $T_{\mathbf{I}_d}\mathbb{S}_d^+$ extended to $\mathbb{S}_d^+$ by left or right multiplication is a bi-invariant metric, where $\mathbf{I}_d$ denotes identity. Equipped with the bi-invariant metrics, geodesics on $\mathbb{S}_d^+$ are simply given by the translated version of the geodesics through the identity element. With some derivations, it can be shown that the distance between two SPD matrices on $\mathbb{S}_d^+$ is

$$d(\mathbf{S}_1, \mathbf{S}_2) = \| \log_{\mathbf{S}_1} \mathbf{S}_2 \|_{\mathbf{S}_1} = \| \log(\mathbf{S}_2) - \log(\mathbf{S}_1) \|,$$

where $\log(\cdot)$ is the normal matrix logarithm. Bi-invariant metrics on $\mathbb{S}_d^+$ are called Log Euclidean metric as they correspond to Euclidean metrics in the logarithmic domain.

2.2. Distance Metric Learning on SPD Matrices

The success of many visual recognition tasks (e.g., image retrieval and categorisation) largely depends on good distance metrics that reflect human perception. Distance metric learning plays an important role in this regard. It aims at learning a distance or similarity metric that keeps the data from the same class close and separates the data from different classes far apart. This often involves the learning of a Mahalanobis matrix $\mathbf{A}$ so that the distance between two data samples $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ is evaluated as $dist_{\mathbf{A}}(\mathbf{x}, \mathbf{y}) = (\mathbf{x} - \mathbf{y})^\top \mathbf{A} (\mathbf{x} - \mathbf{y})$. That is, each data sample $\mathbf{x}$ can be regarded as being transformed as $\widetilde{\mathbf{L}}^\top \mathbf{x}$ and then a Euclidean distance is applied, where $\mathbf{A} = \widetilde{\mathbf{L}}\widetilde{\mathbf{L}}^\top$. A variety of criteria have been proposed to learn $\mathbf{A}$ from data, such as the logDet divergence [2, 19, 9], the max-margin criterion [26] and k-NN classification [27], etc. Nevertheless, when the classic distance metric learning methods extend to SPD matrices, the Euclidean metric used in their design becomes inadequate due to the Riemannian geometry of SPD matrices. A common solution to this problem is to utilise the Log Euclidean metric by converting the geodesic distance of two SPD matrices to the Euclidean distance in their logarithm domain. For example, in [19] (denoted as LE-ITML in this paper), the logarithm of a $d \times d$ SPD matrix $\mathbf{X}$ is unfolded into $\text{vec}(\log \mathbf{X})$, where $\log(\cdot)$ indicates the matrix logarithm and $\text{vec}(\cdot)$ denotes

the vectorisation of a matrix. Then ITML (Information-Theoretic Metric Learning) [2] is exploited to learn a Mahalanobis matrix $\mathbf{A}$ based on the distances $dist_{\mathbf{A}}(\mathbf{X},\mathbf{Y}) = (\text{vec}(\log\mathbf{X}) - \text{vec}(\log\mathbf{Y}))^\top\mathbf{A}(\text{vec}(\log\mathbf{X}) - \text{vec}(\log\mathbf{Y}))$. It can be easily seen that the size of the matrix $\mathbf{A}$ to learn is at $\mathcal{O}(d^4)$. This could lead to a very long learning process and cause overfitting when the number of training samples is not large enough, making the distance metric learning with the original SPD matrices intractable. The work in [9] takes a different approach (denoted as LEML in this paper). It directly learns a transformation function on the square matrix of $\log(\mathbf{X})$, i.e., $f(\log(\mathbf{X})) = \mathbf{W}^\top\log(\mathbf{X})\mathbf{W}$, where $\mathbf{W} \in \mathbb{R}^{d\times k}$ ($k \leq d$). In this way, LEML projects $\mathbf{X}$ from the current tangent space to a new one that better separates different classes. Through the transformation $\mathbf{W}_{d\times k}$, the dimension of $\mathbf{X}$ is reduced to $k\times k$, and these resulting SPD matrices no long maintain the original physical meaning of features.

3. Proposed method

In this section, our proposed metric learning method is first described and followed by the optimisation algorithm and implementation issues. After that, a theoretical analysis of the relationship between the proposed method and the aforementioned two metric learning methods is presented.

We propose to parameterise the SPD matrices with the power (denoting as $\boldsymbol{\alpha}$) of their eigenvalues, and then optimise $\boldsymbol{\alpha}$ through distance metric learning with the Log Euclidean metric. We call this method $\boldsymbol{\alpha}$-CML ($\boldsymbol{\alpha}$-based Covariance-like Metric Learning). As previously mentioned, an SPD matrix is fully characterised by its eigenvalues and eigenvectors. Here we focus on tuning the eigenvalues because the Log Euclidean metric is invariant under unary transformations. For Log Euclidean metric, it is not difficult to show that $d(\mathbf{X},\mathbf{Y}) = d(\mathbf{W}^\top\mathbf{X}\mathbf{W},\mathbf{W}^\top\mathbf{Y}\mathbf{W})$, where $\mathbf{X}$ and $\mathbf{Y}$ are two SPD matrices, while $\mathbf{W}$ is a unary matrix used to map between eigenvector sets. That is, modifying the eigenvector sets of these SPD matrices to another set of eigenvectors via $\mathbf{W}$ will not change the Log Euclidean distance among them. Our algorithm is described as follows.

3.1. Log Euclidean Distance Parameterised by $\boldsymbol{\alpha}$

Given two SPD matrices $\mathbf{X}$ and $\mathbf{Y}$, the Log Euclidean distance between them is defined as:

$$d(\mathbf{X},\mathbf{Y}) = \| \log(\mathbf{X}) - \log(\mathbf{Y}) \|_{\mathcal{F}}, \tag{1}$$

where $\|\cdot\|_{\mathcal{F}}$ denotes the Frobenius norm of a matrix.

Performing eigen-decomposition on $\mathbf{X}$ and $\mathbf{Y}$ leads to $\mathbf{X} = \mathbf{U}_x\mathbf{D}_x\mathbf{U}_x^\top$ and $\mathbf{Y} = \mathbf{U}_y\mathbf{D}_y\mathbf{U}_y^\top$. Here $\mathbf{U} = [\mathbf{u}_1,\mathbf{u}_2,\cdots,\mathbf{u}_d]$ is a $d\times d$ matrix whose column $\mathbf{u}_i$ corresponds to the eigenvectors, and $\mathbf{D}$ is a diagonal matrix whose diagonal element λ_i corresponds to the eigenvalue. It is assumed that for each of the involved SPD matrices, their eigenvalues have been sorted in descending order, and the eigenvectors are also arranged accordingly. We define a parameter vector $\boldsymbol{\alpha} = [\alpha_1,\alpha_2,\cdots,\alpha_d]^\top$ and *use α_i as the power of the ith eigenvalue of each of the SPD matrices*. The resulting matrix is compactly denoted by $\mathbf{X}(\boldsymbol{\alpha}) = \mathbf{U}_x\mathbf{D}_x^{\boldsymbol{\alpha}}\mathbf{U}_x^\top$, where $\mathbf{D}_x^{\boldsymbol{\alpha}}$ is a diagonal matrix whose diagonal is $(\lambda_1^{\alpha_1},\lambda_2^{\alpha_2},\cdots,\lambda_d^{\alpha_d})$. Defining $\boldsymbol{\Lambda}$ to be a diagonal matrix with $\text{diag}(\boldsymbol{\Lambda}) = \boldsymbol{\alpha}$, it is not difficult to show

$$\begin{aligned}\log(\mathbf{X}(\boldsymbol{\alpha})) &= \mathbf{U}_x\boldsymbol{\Lambda}\log[\mathbf{D}_x]\mathbf{U}_x^\top \equiv \mathbf{U}_x\boldsymbol{\Lambda}\mathbf{E}_x\mathbf{U}_x^\top,\\ \log(\mathbf{Y}(\boldsymbol{\alpha})) &= \mathbf{U}_y\boldsymbol{\Lambda}\log[\mathbf{D}_y]\mathbf{U}_y^\top \equiv \mathbf{U}_y\boldsymbol{\Lambda}\mathbf{E}_y\mathbf{U}_y^\top,\end{aligned} \tag{2}$$

where $\log[\mathbf{D}_x]$ denotes the diagonal matrix obtained after applying the natural logarithm to the diagonal elements of $\mathbf{D}_x$ (The square bracket $[\cdot]$ is used to differentiate it from the matrix logarithm). Note that the last step is because we define $\mathbf{E}_x \equiv \log[\mathbf{D}_x]$ for the sake of clarity, and that $\boldsymbol{\Lambda}\mathbf{E}_x = \mathbf{E}_x\boldsymbol{\Lambda}$ because both are diagonal. By Eqn. (2), we immediately have the following result on trace

$$\begin{aligned}\text{trace}(\log^\top(\mathbf{X}(\boldsymbol{\alpha}))\log(\mathbf{X}(\boldsymbol{\alpha}))) &= \boldsymbol{\alpha}^\top\mathbf{E}_x^2\boldsymbol{\alpha},\\ \text{trace}(\log^\top(\mathbf{Y}(\boldsymbol{\alpha}))\log(\mathbf{Y}(\boldsymbol{\alpha}))) &= \boldsymbol{\alpha}^\top\mathbf{E}_y^2\boldsymbol{\alpha}.\end{aligned} \tag{3}$$

For the cross-term, it can be derived as follows

$$\begin{aligned}&\text{trace}(\log^\top(\mathbf{X}(\boldsymbol{\alpha}))\log(\mathbf{Y}(\boldsymbol{\alpha})))\\ =\ &\text{trace}(\mathbf{U}_x\mathbf{E}_x\boldsymbol{\Lambda}\mathbf{U}_x^\top\mathbf{U}_y\boldsymbol{\Lambda}\mathbf{E}_y\mathbf{U}_y^\top)\\ =\ &\text{trace}(\boldsymbol{\Lambda}\underline{\mathbf{U}_x^\top\mathbf{U}_y}\boldsymbol{\Lambda}\mathbf{E}_y\underline{\mathbf{U}_y^\top\mathbf{U}_x}\mathbf{E}_x)\\ &(\text{Define } \mathbf{W}_{xy} = \mathbf{U}_x^\top\mathbf{U}_y)\\ =\ &\text{trace}(\boldsymbol{\Lambda}\mathbf{W}_{xy}\boldsymbol{\Lambda}\underline{\mathbf{E}_y\mathbf{W}_{xy}^\top\mathbf{E}_x})\\ &(\text{Define } \mathbf{B}_{xy} = \mathbf{E}_y\mathbf{W}_{xy}^\top\mathbf{E}_x)\\ \equiv\ &\text{trace}(\boldsymbol{\Lambda}\mathbf{W}_{xy}\boldsymbol{\Lambda}\mathbf{B}_{xy})\\ =\ &\boldsymbol{\alpha}^\top\mathbf{C}_{xy}\boldsymbol{\alpha}.\end{aligned} \tag{4}$$

In the last step, we define $(\mathbf{C}_{xy})_{ij} = (\mathbf{W}_{xy})_{ij}(\mathbf{B}_{xy})_{ij}$, where the subscript ij denotes the ij-th element in the matrix. Combining the above results, the Log Euclidean distance between $\mathbf{X}(\boldsymbol{\alpha})$ and $\mathbf{Y}(\boldsymbol{\alpha})$ can be computed as

$$\begin{aligned}d(\mathbf{X}(\boldsymbol{\alpha}),\mathbf{Y}(\boldsymbol{\alpha})) =& \| \log(\mathbf{X}(\boldsymbol{\alpha})) - \log(\mathbf{Y}(\boldsymbol{\alpha})) \|_{\mathcal{F}}^2\\ =& \boldsymbol{\alpha}^\top(\mathbf{E}_x^2 - 2\mathbf{C}_{xy} + \mathbf{E}_y^2)\boldsymbol{\alpha}\\ \equiv& \boldsymbol{\alpha}^\top\mathbf{M}_{xy}\boldsymbol{\alpha},\end{aligned} \tag{5}$$

where $\mathbf{M}_{xy}$ is defined as $\mathbf{E}_x^2 - 2\mathbf{C}_{xy} + \mathbf{E}_y^2$.

3.2. Distance Metric Learning via $\boldsymbol{\alpha}$

For the sake of clarity, in the following, we denote the matrix $\mathbf{M}_{xy}$ as $\mathbf{M}(\mathbf{X},\mathbf{Y})$ to explicitly show its dependency on the two input matrices. Let $\{i,j,k\}$ be the index of a

triplet $\{\mathbf{X}_i, \mathbf{X}_j, \mathbf{X}_k\}$, where $\mathbf{X}_i$ and $\mathbf{X}_j$ belong to the same class while $\mathbf{X}_i$ and $\mathbf{X}_k$ belong to different classes. Each index $\{i, j, k\}$ corresponds to an integer $p \in \mathcal{S}$, where $\mathcal{S}$ contains all possible triplets. A max-margin based distance metric learning is developed for SPD matrices as follows. Let r be the margin and ξ be the slack variable. Given a triplet $\{i, j, k\}$, the Log Euclidean distance of $(\mathbf{X}_i, \mathbf{X}_k)$ shall be greater than that of $(\mathbf{X}_i, \mathbf{X}_j)$ by a margin if possible. The metric learning is then formulated as a constrained margin-maximisation problem as follows,

$$\begin{aligned} \text{P1:} \max_{\boldsymbol{\alpha}, \xi_p, r} \; & r - \lambda \sum_{p \in \mathcal{S}} \xi_p \\ s.t. \; & \boldsymbol{\alpha}^\top \mathbf{M}(\mathbf{X}_i, \mathbf{X}_k) \boldsymbol{\alpha} - \boldsymbol{\alpha}^\top \mathbf{M}(\mathbf{X}_i, \mathbf{X}_j) \boldsymbol{\alpha} \geq r - \xi_p, \\ & \boldsymbol{\alpha}^\top \boldsymbol{\alpha} = 1, \;\; r \geq 0, \;\; \xi_p \geq 0, \quad \forall p \in \mathcal{S}. \end{aligned} \tag{6}$$

Denoting $\Delta\mathbf{M}_p = \mathbf{M}(\mathbf{X}_i, \mathbf{X}_k) - \mathbf{M}(\mathbf{X}_i, \mathbf{X}_j)$ and converting maximisation to minimisation, Eqn. (6) becomes

$$\begin{aligned} \text{P1:} \min_{\boldsymbol{\alpha}, \xi_p, r} \; & \lambda \sum_{p \in \mathcal{S}} \xi_p - r \\ s.t. \; & \boldsymbol{\alpha}^\top \Delta\mathbf{M}_p \boldsymbol{\alpha} \geq r - \xi_p, \\ & \boldsymbol{\alpha}^\top \boldsymbol{\alpha} = 1, \;\; r \geq 0, \;\; \xi_p \geq 0, \quad \forall p \in \mathcal{S}. \end{aligned} \tag{7}$$

Furthermore, noting that $\boldsymbol{\alpha}^\top \Delta\mathbf{M}_p \boldsymbol{\alpha} = \text{tr}(\Delta\mathbf{M}_p \boldsymbol{\alpha}\boldsymbol{\alpha}^\top)$ and defining $\mathbf{L} = \boldsymbol{\alpha}\boldsymbol{\alpha}^\top$, optimising the problem P1 is equal to optimising the following problem P2.

$$\begin{aligned} \text{P2:} \min_{\mathbf{L} \succeq 0, \xi_p, r} \; & \lambda \sum_{p \in \mathcal{S}} \xi_p - r \\ s.t. \; & \langle \mathbf{L}, \Delta\mathbf{M}_p^\top \rangle_{\mathcal{F}} \geq r - \xi_p, \quad \forall p \in \mathcal{S} \\ & \text{trace}(\mathbf{L}) = 1, \;\; r \geq 0, \;\; \xi_p \geq 0, \;\; \text{rank}(\mathbf{L}) = 1. \end{aligned} \tag{8}$$

By definition, $\mathbf{L}$ is constrained to be semi-definite positive and a rank-one matrix. The symbol $\langle \cdot, \cdot \rangle_{\mathcal{F}}$ denotes the inner product under the Frobenius norm. That is, $\langle \mathbf{L}, \Delta\mathbf{M}_p^\top \rangle_{\mathcal{F}} = \text{trace}(\Delta\mathbf{M}_p \mathbf{L})$, which is a linear function of $\mathbf{L}$. By some rearrangement, the optimisation problem P2 can be further rewritten as the follows.

$$\begin{aligned} \text{P3:} \min_{\mathbf{L} \succeq 0, r} \; & -r + \lambda \sum_{p \in \mathcal{S}} \ell\left(\langle \mathbf{L}, \Delta\mathbf{M}_p^\top \rangle_{\mathcal{F}} - r\right) \\ s.t. \; & \text{trace}(\mathbf{L}) = 1, \;\; r \geq 0, \;\; \text{rank}(\mathbf{L}) = 1, \end{aligned} \tag{9}$$

where $\ell(z) = \max(0, -z)$ is the hinge loss function.

3.3. Optimisation

The optimisation problem P3 can be efficiently solved by following the projected gradient descent framework. The key issue is how to deal with the constraint that $\mathbf{L}$ is a rank-one SPD matrix. It is known that each SPD matrix can be decomposed as a linear convex combination of a set of rank-one matrices. In light of the above rule, at each iteration, $\mathbf{L}$ is updated by

$$\mathbf{L}^{t+1} = (1 - \beta)\mathbf{L}^t + \beta\Delta\mathbf{L}, \tag{10}$$

where $\Delta\mathbf{L}$ (and $\mathbf{L}^t$ initialised at $t = 0$) is a rank-one and trace-one matrix. Requiring the trace of $\Delta\mathbf{L}$ to be one avoids the scaling problem of the optimisation. The parameter $0 \leq \beta \leq 1$ determines the update step, obtained by following the line search algorithm used in the gradient descent optimisation. Note that $\mathbf{L}^{t+1}$ obtained by Eqn. (10) is not guaranteed to be rank-one. However, because $\mathbf{L}^t$ is constrained to be rank-one, $\mathbf{L}^{t+1}$ at most has the rank of two, which, when projected into the rank-one matrix set, will not cause much loss.

The optimal $\Delta\mathbf{L}$ at each iteration is sought as follows. Let $f(\mathbf{L}, r) = -r + \lambda \sum_{p \in \mathcal{S}} \ell\left(\langle \mathbf{L}, \Delta\mathbf{M}_p^\top \rangle_{\mathcal{F}} - r\right)$ and $\bigtriangledown f$ denote the gradient matrix of f with the fixed r. Obviously, in order to maximally decrease the objective function value with the update rule in Eqn. (10), the optimal $\Delta\mathbf{L}$ should best approximate the negative gradient matrix $\bigtriangledown f$, i.e.,

$$\begin{aligned} \Delta\mathbf{L}^\star = \arg \max_{\Delta\mathbf{L}} & \langle -\bigtriangledown f(\mathbf{L}, r), \Delta\mathbf{L} \rangle, \\ s.t. \quad & \text{rank}(\Delta\mathbf{L}) = 1, \;\; \text{trace}(\Delta\mathbf{L}) = 1. \end{aligned} \tag{11}$$

It is not difficult to show that $\Delta\mathbf{L}^\star = \mathbf{v}\mathbf{v}^\top$, where $\mathbf{v}$ is the leading eigenvector of the matrix $-\bigtriangledown f(\mathbf{L}, r)$, corresponding to its largest eigenvalue. Therefore, optimising $\mathbf{L}$ in fact boils down to computing a d-dimensional leading eigenvector $\mathbf{v}$. And this can be much more efficient than solving the whole set of eigenvectors. The complete algorithm is summarised in Algorithm 1.

3.4. Implementation Issues

We follow the work in [26] to generate the triplets for training. For a given sample $\mathbf{X}_i$, firstly its t nearest neighbours in the same class are identified (called positive neighbours). Then for each positive neighbour, the samples from different classes (called negative neighbours) are picked out if they are closer to $\mathbf{X}_i$ than this positive neighbour. The combinations of $\mathbf{X}_i$ and its positive and negative neighbours form the triplets. This may lead to up to tens of thousands of triplets. When used to calculate $\bigtriangledown f(\mathbf{L}, r)$ at each iteration of gradient descent, the large number of triplets may incur expensive computational cost. To speed up the optimisation, we employ the following strategy. Note that $\bigtriangledown f(\mathbf{L}, r) = -\lambda \sum_{p \in \mathcal{S}'} \Delta\mathbf{M}_p^\top$, where $\mathcal{S}'$ denotes the set of indices of triplets that violate the margin. Therefore, we initially calculate and store the value of $\sum_{p \in \mathcal{S}} \Delta\mathbf{M}_p^\top$ using all triplets. Then in each iteration, we only evaluate $\Delta\mathbf{M}_p^\top$ for the changing triplets, i.e., the newly violated triplets and the newly satisfied triplets caused by the last iteration of optimisation. Although the number of initial triplets may be

Algorithm 1 Distance Metric Learning with $\boldsymbol{\alpha}$-CML

Input: a training sample set $\boldsymbol{\Omega} = \{(\mathbf{X}_i, y_i)\}_{i=1}^n$, (where $\mathbf{X}_i \in \mathbb{R}^{d\times d}$ and y_i is class label) and a set of triplets.

1. Initialize $\mathbf{L}^0 = \mathbf{1}\mathbf{1}^\top/d$, where $\mathbf{1}$ is d-dimensional vector with all elements to be one.
2. Let $\mathbf{L} = \mathbf{L}^0$, solve P3 to obtain r^0.
3. Set t = 0.

repeat

4. Calculate $(-\bigtriangledown f(\mathbf{L}, r))$ in Section 3.3 and its eigenvector $\mathbf{v}$ corresponding to the largest eigenvalue.
5. Update $\mathbf{L}^{t+1}$ according to Eqn. (10); $\Delta\mathbf{L}$ is obtained by $\mathbf{v}\mathbf{v}^\top$; β is obtained by line search.
6. Project $\mathbf{L}^{t+1}$ back to be a rank-one matrix.
7. Update $\mathbf{r}^{t+1}$ by solving P3 with $\mathbf{L}^{t+1}$ fixed.
8. Set t = t+1

until convergence or the maximum number of iterations.

Output: $\mathbf{L}^\star = \mathbf{L}^{(t)}$, $\boldsymbol{\alpha}^\star$ is the eigenvector of $\mathbf{L}^\star$ corresponding to the largest eigenvalue.

large, the number of changing triplets is much smaller, usually less than 100, at each iteration. In this way, the optimisation can be efficiently solved. Typically, a desktop computer with 3.0GHz CPU and 8.0G RAM takes 0.07s for one iteration (not the initial one) of optimisation on 128 43×43 SPD matrices with about $3,000$ triplets in the experiment.

3.5. Relationship to other SPD-ML methods

The proposed $\boldsymbol{\alpha}$-CML and another two methods in [9, 19] represent three different members in the family of metric learning methods for SPD matrix. It is desirable to manifest their relationship to gain more insights on the connections and differences.

3.5.1 Link to LEML in [9]

As indicated in Eqn.(5) in [9], the goal of LEML is to learn a linear transformation $\mathbf{W} \in \mathbb{R}^{d\times k}$ applied to the logarithm of a SPD matrix $\mathbf{X}_{d\times d}$,

$$f(\log(\mathbf{X})) = \mathbf{W}^\top \log(\mathbf{X})\mathbf{W}. \tag{12}$$

Recall that the eigen-decomposition of $\mathbf{X}$ is $\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{U}^\top$ and that $\log(\mathbf{X}) = \mathbf{U}\log[\mathbf{D}]\mathbf{U}^\top$. The proposed method $\boldsymbol{\alpha}$-CML applies a vector $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \cdots, \alpha_d]^\top$ to the (sorted) eigenvalues of $\mathbf{X}$, respectively, and the resulting matrix was denoted by $\mathbf{X}(\boldsymbol{\alpha})$. Also, as previously defined, $\boldsymbol{\Lambda}$ is a diagonal matrix whose diagonal is $\boldsymbol{\alpha}$. By some manipulation, we can "fit" $\boldsymbol{\alpha}$-CML into the form of LEML as

$$\begin{aligned}
\log(\mathbf{X}(\boldsymbol{\alpha})) &= \mathbf{U}\boldsymbol{\Lambda}\log[\mathbf{D}]\mathbf{U}^\top \\
&= \mathbf{U}\boldsymbol{\Lambda}^{\frac{1}{2}}\log[\mathbf{D}]\boldsymbol{\Lambda}^{\frac{1}{2}}\mathbf{U}^\top \\
&= (\mathbf{U}\boldsymbol{\Lambda}^{\frac{1}{2}}\mathbf{U}^\top)(\mathbf{U}\log[\mathbf{D}]\mathbf{U}^\top)(\mathbf{U}\boldsymbol{\Lambda}^{\frac{1}{2}}\mathbf{U}^\top) \\
&= (\mathbf{U}\boldsymbol{\Lambda}^{\frac{1}{2}}\mathbf{U}^\top)^\top \log(\mathbf{X})(\mathbf{U}\boldsymbol{\Lambda}^{\frac{1}{2}}\mathbf{U}^\top) \\
&\quad (\text{Define} \quad \mathbf{W}_x \equiv \mathbf{U}\boldsymbol{\Lambda}^{\frac{1}{2}}\mathbf{U}^\top) \\
&= \mathbf{W}_x^\top \log(\mathbf{X})\mathbf{W}_x.
\end{aligned} \tag{13}$$

Therefore, the above result indicates that $\boldsymbol{\alpha}$-CML can also be interpreted as applying a linear transformation $\mathbf{W}$ to $\log(\mathbf{X})$ as in LEML, and this shows the connection.

However, a **significant difference** lies at that $\mathbf{W}_x$ in $\boldsymbol{\alpha}$-CML varies with $\mathbf{X}$, because the eigenvector matrix $\mathbf{U}$ depends on $\mathbf{X}$. From the perspective of learning a linear transformation, $\boldsymbol{\alpha}$-CML learns a transformation that is **sample-specific**. This is in contrast with (and cannot be directly achieved via) the LEML method, which learns a fixed transformation $\mathbf{W}$ equally applied to all samples. It is observed in the experiment that this sample-specific property could even help $\boldsymbol{\alpha}$-CML to win LEML in some situation, although the total number of parameters is one-order less.

3.5.2 Link to LE-ITML in [19]

Following the above notations, we write the eigenvector and eigenvalue matrices in full as $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_d]$ and $\mathbf{D} = \text{diag}(\lambda_1, \lambda_2, \cdots, \lambda_d)$. And it can be immediately obtained that $\log(\mathbf{X}) = \mathbf{U}\log[\mathbf{D}]\mathbf{U}^\top = \sum_{i=1}^d (\log\lambda_i)\mathbf{u}_i\mathbf{u}_i^\top$. Recalling that $\text{vec}(\cdot)$ denotes the vectorisation of a matrix, it is trivial to show that $\text{vec}(\log(\mathbf{X})) = \sum_{i=1}^d \mathbf{v}_i$, where $\mathbf{v}_i \equiv \text{vec}((\log\lambda_i)\mathbf{u}_i\mathbf{u}_i^\top)$ and $\mathbf{v}_i \in \mathbb{R}^{d^2}$.

Proposition. *Recall that $\boldsymbol{\alpha}$ is the power of the eigenvalues of $\mathbf{X}$, i.e., $\mathbf{X}(\boldsymbol{\alpha}) = \mathbf{U}\mathbf{D}^{\boldsymbol{\alpha}}\mathbf{U}^\top$. It can be shown that $vec(\log(\mathbf{X}(\boldsymbol{\alpha}))) = \boldsymbol{\Gamma}\boldsymbol{\alpha}$, where $\boldsymbol{\Gamma} = [\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_d]$ and its columns form a set of d orthogonal bases spanning a d-dimensional subspace $\mathcal{V}$ in the whole space of $\mathbb{R}^{d^2}$.* (Proof is provided in the supplement)

By this Proposition, it is known that $\text{vec}(\log(\mathbf{X}(\boldsymbol{\alpha})))$ corresponds to a point $A_{\boldsymbol{\alpha}} = (\alpha_1, \alpha_2, \cdots, \alpha_d)^\top$ in the subspace $\mathcal{V}$ spanned by $\mathbf{v}_i$ ($i = 1, \cdots, d$). Also, because $\text{vec}(\log(\mathbf{X})) = \sum_{i=1}^d \mathbf{v}_i = \boldsymbol{\Gamma}\mathbf{1}$, it corresponds to point $A_0 = (1, 1, \cdots, 1)^\top$ in $\mathcal{V}$. In this way, the proposed $\boldsymbol{\alpha}$-CML can be interpreted as learning a mapping from point A_0 to $A_{\boldsymbol{\alpha}}$ in the subspace $\mathcal{V}$. As for LE-ITML [19], it applies a linear transformation $\widetilde{\mathbf{L}}$ ($\widetilde{\mathbf{L}} \in \mathbb{R}^{d^2\times p}, p \le d^2$) to $\text{vec}(\log(\mathbf{X}))$ as

$$\widetilde{\mathbf{L}}^\top \text{vec}(\log(\mathbf{X})) = \widetilde{\mathbf{L}}^\top(\mathbf{v}_1 + \cdots + \mathbf{v}_d). \tag{14}$$

Similarly, this can be interpreted as mapping point A_0 in $\mathcal{V}$ to another point $A_{\widetilde{\mathbf{L}}}$. However, the **difference** lies at that

because $\widetilde{\mathbf{L}}$ can be any $d^2 \times p$ matrix, the resulting point $A_{\widetilde{\mathbf{L}}}$ does not necessarily reside in the subspace $\mathcal{V}$ (as the case in $\boldsymbol{\alpha}$-CML) any more, but could locate anywhere in the whole space $\mathbb{R}^{d^2}$ instead. This brings higher learning capability to LE-ITML. However, the price is a three-order-larger number of parameters and the scalability issue with increasing dimensions.

Before the end of the analysis, it is worth noting that the proposed method $\boldsymbol{\alpha}$-CML adjusts the SPD matrices in the original space and does not involve any dimension reduction or projection. Therefore, the methods like LEML and LE-ITML can still be applied after $\boldsymbol{\alpha}$-CML to pursue greater discrimination in further. In this sense, the proposed $\boldsymbol{\alpha}$-CML does not compete with, but complement, the existing distance metric learning methods for SPD matrices.

4. Experimental Result

We test the proposed metric learning method on three types of recognition tasks: texture image classification, human action recognition, and brain image analysis. They cover different types of SPD-based representations. Specifically, the first task uses covariance matrix as feature representation; the second one uses non-linear kernel matrix as feature representation; and the last one uses inverse covariance matrix as feature representation. Example images of these data sets are given in Fig. 1. Following the tradition of distance metric learning, a k-nearest-neighbour classifier using the Log Euclidean distance is employed to assess the performance of the proposed method.

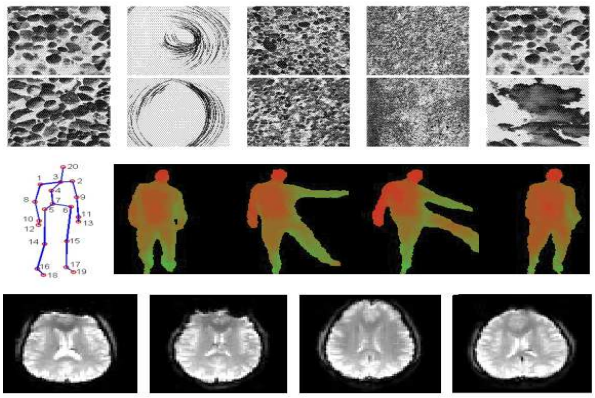

Figure 1. *Example images from the used data sets. Top: five most difficult pairs of Brodatz data set. Middle: employed skeleton and example actions from MSR-Action3D data set. Bottom: four examples of resting-state fMRI images from ADHD-200 data set.*

4.1. Result on texture image classification

The Brodatz texture data set has been commonly used in the literature to evaluate algorithms using covariance-based representation. This data set contains 112 images, each corresponding to one class of texture. Following the work in [6], 64 sub-regions are cropped from each image as the samples of the corresponding texture class. For each sub-region, a 43-dimensional vector (including image intensity and 2D Gabor wavelets) is extracted at each pixel, based on which, the covariance descriptor for each sample is constructed. Both binary classification and multi-class classification are tested on this data set in the experiment.

For binary classification, 15 pairs of classes that are most difficult to discriminate from each other (identified as having the lowest accuracy generated by a k-NN classifier. See the supplement for the class labels) are selected for test. The remaining pairs are not included because almost 100% accuracy can be achieved on them. For the proposed method $\boldsymbol{\alpha}$-CML, its parameter t (the number of positive neighbours) used in triplet generation is set as either 3 or 5, and the parameter λ in the optimisation problem P3 to balance the margin and loss is set as either 1.0 or 3.0, due to the variation of different binary-class groups. As each pair has only 128 (64×2) samples in total, the Leave-One-Out (LOO) strategy is employed to make full use of samples for training, and the LOO classification accuracy is reported using the k-NN classifier with $k = 1, 3, 5$ and 7.

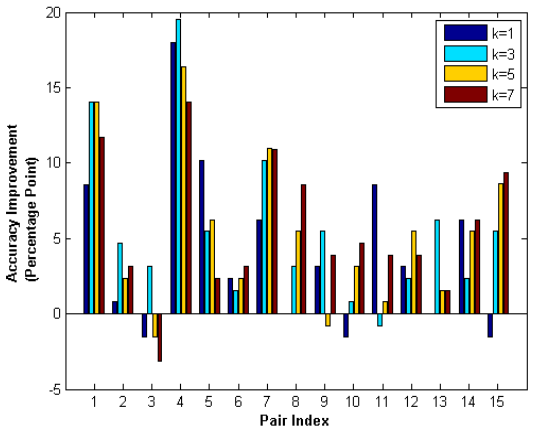

Figure 2. Brodatz binary classification: accuracy improvement (in percentage point) after applying the proposed method $\boldsymbol{\alpha}$-CML.

To verify whether the proposed method can produce a better metric and therefore higher classification performance, we compare the LOO classification accuracy obtained with and without applying the proposed method. Fig. 2 shows the improvement (in percentage point) achieved on each of the 15 pairs under different k values. As seen, metric learning with the proposed method is quite promising, and improvement is observed on 14 pairs. The magnitude can reach more than 10 percentage point on four pairs, five percentage points on 11 pairs, and the highest one reaches almost 20 percentage points. At the same time, Pair 3 shows decreased accuracies when $k = 1, 5, 7$. However, checking the result of $k = 3$ (the parameter t in the

proposed method is set as 3 for Pair 3), we still observe clear improvement after metric learning. The average accuracy improvements over all the 15 pairs are given in Table 1, and paired t-test is conducted to verify the statistical significance of the improvement. As seen, all of them are statistically significant at the level of 0.05 (indicated by the p-value smaller than 0.05). In addition, it is interesting to see that we use $t = 3$ or $t = 5$ to generate triplets, yet the improvement is consistently observed at most of the k values (except the case in Pair 3). This can be viewed as a good indicator that a better metric has indeed been learned by the proposed method.

Table 1. Brodatz binary classification: average accuracy improvement (in percentage point) over all the 15 pairs achieved by the proposed method $\boldsymbol{\alpha}$-CML.

%	k=1	k=3	k=5	k=7
Mean ↑	4.2 ± 5.5	5.6 ± 5.3	5.4 ± 5.2	5.6 ± 4.5
p-value	1e-2	1e-3	1e-3	3e-4

For the case of multi-class classification, all 112 texture classes (7168 samples in total) are considered. The data set is randomly split into two equal-sized subsets for training and test, and this procedure is repeated for 10 times to report the average classification performance. The parameter t is set as 5, and λ is set as 5.0. The average accuracy improvements are reported in Table 2. Similar to the situation of binary classification, statistically significant improvement of accuracy is consistently observed on all k values, demonstrating the effectiveness of the proposed method again.

Table 2. Brodatz multi-class classification: average accuracy improvement (in percentage point) over all the 15 pairs achieved by the proposed method $\boldsymbol{\alpha}$-CML.

%	k=1	k=3	k=5	k=7
Mean ↑	2.5 ± 1.0	2.8 ± 1.1	2.6 ± 1.0	2.4 ± 0.8
p-value	2e-5	3e-5	2e-5	4e-6

4.2. Result on human action recognition

MSR-Action3D contains 20 actions performed by 10 subjects. Each action is done two or three times by each subject. Only the skeletal data is used in our experiment. Each action instance contains $40 \sim 60$ frames, and each frame is represented by 120-dimensional feature vector corresponding to the coordinate differences of 3D skeleton points between a frame and its two neighbouring frames. Following the literature [22], we use a Gaussian RBF kernel matrix based SPD representation (called Ker-RP-RBF in that work) for each action instance, as this representation reported the state-of-the-art performance on this data set. We want to investigate whether applying the proposed method can boost this performance in further. To facilitate the comparison, we follow the literature to use the odd-indexed subjects for training and the even-indexed ones for test. As previous, the parameter t used for triplets generation is set as 3, and the parameter λ is set as 1.0. The performance of various methods on this data set is quoted in Table 3 for comparison.

The performance of the proposed method is first tested by k-NN classifier and the result ($k = 1$) is given in Table 3. As shown, an improvement of 4.6 percentage point is brought by the proposed $\boldsymbol{\alpha}$-CML after learning on Ker-RP-RBF with kNN classifier. This result (92.7%) is better than that of most methods in comparison except [22, 30, 28]. To directly compare with the state-of-the-art result on this data set in [22], an SVM classifier with the Log Euclidean kernel is used for test by following that work. Note that the Log Euclidean kernel is now calculated by using the Ker-RP-RBF adjusted by the proposed method $\boldsymbol{\alpha}$-CML. As shown, the Ker-RP-RBF in [22] achieves an accuracy of 96.1%[2] with an SVM classifier, while our proposed method further improves this result to 97.3%, higher than all the other methods. This result again verifies the effectiveness of the proposed metric learning method. In addition, it is interesting to note that although metric learning is usually applied to k-NN classification, learning a good metric could help kernel evaluation and in turn improve the performance of SVM classification, as shown by this experiment.

Table 3. Comparison of classification accuracy (in percentage point) on MSR-Action3D data set.

Compared methods	ACC (in %)
Pose Set [21]	90.0
Hierarchy of Cov3DJs [10]	90.5
Moving Pose [32]	91.7
Lie Group [18]	92.5
SNV [30]	93.1
Spatiotemp. Features Fusing [28]	94.3
Cov-RP [17]	74.0
Cov-$J_{\mathcal{H}}$-SVM [6]	80.4
Ker-RP-RBF+kNN	88.1
Ker-RP-RBG+$\boldsymbol{\alpha}$-CML+kNN (proposed)	92.7
Ker-RP-RBF+SVM [22]	96.1
Ker-RP-RBF+$\boldsymbol{\alpha}$-CML+SVM (proposed)	**97.3**

4.3. Results on brain image classification

ADHD-200 is a publicly accessible data set provided by the Neuro Bureau for the prediction of Attention Deficit Hyperactivity Disorder (ADHD). It consists of the resting-state fMRI images of 768 training subjects and 197 test subjects, collected from eight independent imaging sites. To cancel out the variation in image processing steps, we adopt the preprocessed data provided by ADHD-200 using Athena

[2]The result of our implementation is slightly different from that reported in [22], possibly due to the cross-validation choice of parameters.

pipeline. Each brain image is partitioned into 90 cerebral brain regions and each region is characterised by a feature vector corresponding to its mean time-series. There are 26 test samples containing invalid values in the time series, which are removed from the test. In this experiment, following the literature, the inverse covariance representation is employed for each brain image. Both k-NN and SVM classifiers are used to evaluate the classification performance. As shown in Table 4, the proposed $\boldsymbol{\alpha}$-CML achieves an accuracy increase of 4.1% for k-NN and 2.9% for SVM, which strengthens our previous observation. Moreover, the results of using Local Clustering Coefficients (LCC), Stein Kernel (SK), and Cholesky (CHK) kernel with SVMs respectively are also presented for reference. Note that SK and CHK are SPD-matrix based kernels.

Table 4. ADHD-200: Comparison of classification accuracy on the predefined training-test partition

ACC (in %)	k-NN	SVM	Existing methods	ACC (in %)
Original	64.3	66.7	LCC+SVM	64.3
$\boldsymbol{\alpha}$-CML (proposed)	**68.4**	**69.6**	SK+SVM	63.7
Improvement	4.1	2.9	CHK+SVM	63.2

4.4. Comparison with existing SPD-ML methods

Comparison experiment is now conducted between the proposed $\boldsymbol{\alpha}$-CML method and another two SPD-ML methods of LE-ITML and LEML (their codes are obtained from the respective authors' websites). Note that, on the above data sets, LE-ITML needs to solve very large Mahalanobis matrices and could not return the results of any data set after 12 hours, making this method computationally impractical on these scales of data sets (and also very hard to tune its parameters). Therefore, in the following, only the results of LEML and $\boldsymbol{\alpha}$-CML are presented.

Table 5 shows the best accuracies of k-NN classifiers on Brodatz's 15 most difficult pairs. It can be seen that both $\boldsymbol{\alpha}$-CML and LEML could win or lose with the varied data sets, which is somewhat expected. Via the analysis in Section 3.5, we can regard $\boldsymbol{\alpha}$-CML as weighing different features without changing the original feature space, while LEML conducts dimension reduction via projection. To some extent, their relationship could be the analogue to that of feature weighting and dimension reduction. Which one is better is quite data-dependent, just as the case of feature weighting versus dimension reduction.

Table 6 compares the performance of our proposed $\boldsymbol{\alpha}$-CML with the existing methods of LEML, CDL [23], and RSR-ML [5]. Note that CDL and RSR-ML are related to SPD-ML methods from the perspective of learning transformation matrix to project SPD matrices to a lower dimensional space. CDL utilises Log Euclidean metric, while RSR-ML utilises AIM and Stein kernel. For comparison, we additionally test an action dataset HDM05 [5], for which the results of CDL and RSR-ML can be quoted from [5]. For HDM05, LEML and our methods are tested on Ker-RP in addition to COV-RP. Also, RSR-ML is applied on Brodatz, MSR-Action3D and ADHD-200 with the code provided by the authors. The results in Table 6 demonstrate the effectiveness of $\boldsymbol{\alpha}$-CML on various SPD-matrix based visual representations, compared with existing methods. Moreover, the last row in Table 6 gives the results of $\boldsymbol{\alpha}$-CML + LEML for metric learning on the two action recognition data sets HDM05 and MSR-Action3D. The improvement over both methods indicates the potential benefits of combining these two different metric learning schemes.

Table 5. Comparison on Brodatz's 15 most difficult pairs

ACC (in %)	Pair 1	Pair 2	Pair 3	Pair 4	Pair 5
$\boldsymbol{\alpha}$-CML	81.3	**72.7**	66.4	**83.6**	**79.7**
LEML [9]	**82.0**	71.1	**76.6**	70.3	72.7
%	Pair 6	Pair 7	Pair 8	Pair 9	Pair 10
$\boldsymbol{\alpha}$-CML	75.0	**89.1**	**86.7**	**89.1**	**76.6**
LEML [9]	**78.9**	78.9	**86.7**	**89.1**	71.1
%	Pair 11	Pair 12	Pair 13	Pair 14	Pair 15
$\boldsymbol{\alpha}$-CML	**91.4**	**92.2**	86.7	**98.4**	**85.9**
LEML [9]	90.6	90.6	**90.6**	**98.4**	84.4

Table 6. ***Comparison of Methods (in ACC %)***

Data sets	HDM05	MSR-Action3D	Brodatz	ADHD
SPD-RP	COV/Ker-RP	Ker-RP	COV	ICOV
CDL [23]	79.8	N.A.	N.A.	N.A.
RSR-ML [5]	81.9	85.0	73.8	62.6
LEML [9]	89.7/93.1	91.2	**76.4**	67.3
α-CML (proposed)	**91.0/93.6**	**92.7**	74.3	**68.4**
α-CML+LEML (proposed)	**96.6/96.6**	**94.6**	–	–

5. Conclusion

In this paper, we introduce a new member to the family of metric learning for SPD matrices. It owns some interesting properties compared with the existing related methods, e.g., less learning variables and being sample-specific, and has been evaluated on different SPD matrix based representations. Meanwhile, the relationship between the proposed and the existing methods has also been discussed via theoretical analysis. Enriched variants of this model will be studied in our future work, e.g, learning class-specific $\boldsymbol{\alpha}$.

6. Acknowledgement

The authors thank Australian Research Council (ARC DE160100241) and National Science Foundation of China (61432008, 61673203) for the support of this work.

References

[1] V. Arsigny, P. Fillard, X. Pennec, and N. Ayache. Log-euclidean metrics for fast and simple calculus on diffusion tensors. *Magn Reson Med*, 56(2):411–21, 2006. 1, 2

[2] J. Davis, B. Kulis, P. Jain, S. Sra, and I. Dhillon. Information-theoretic metric learning. In *ICML*, 2007. 1, 2, 3

[3] I. Dryden, A. Koloydenko, and D. Zhou. Non-euclidean statistics for covariance matrices, with applications to diffusion tensor imaging. *Annu. Appl. Stat*, 3(3):1102–1123, 2009. 1

[4] W. Forstner and B. Moonen. *A metric for covariance matrices*. Berlin Heidelberg: Springer, 2003. 1

[5] M. Harandi, M. Salzmann, and R. Hartley. From manifold to manifold: Geometry-aware dimensionality reduction for spd matrices. In *ECCV*, pages 17–32, 2014. 8

[6] M. Harandi, M. Salzmann, and F. Porikli. Bregman divergences for infinite dimensional covariance matrices. In *CVPR*, pages 1003–1010, 2014. 6, 7

[7] M. Hayat, M. Bennamoun, and S. An. Reverse training: An efficient approach for image set classification. In *ECCV*, 2014. 1

[8] S. Huang, J. Li, L. Sun, J. Liu, T. Wu, K. Chen, A. Fleisher, E. Reiman, and J. Ye. Learning brain connectivity of alzheimer's disease from neuroimaging data. In *NIPS*, 2009. 1

[9] Z. Huang, R. Wang, S. Shan, X. Li, and X. Chen. Log-euclidean metric learning on symmetric positive definite manifold with application to image set classification. In *ICML*, 2015. 1, 2, 3, 5, 8

[10] M. Hussein, M. Torki, M. Gowayyed, and M. El-Saban. Human action recognition using a temporal hierarchy of covariance descriptors on 3d joint locations. In *IJCAI*, 2013. 1, 7

[11] A. Mahmood, A. Mian, and R. Owens. Semi-supervised spectral clustering for image set classification. In *CVPR*, 2014. 1

[12] S. Paisitkriangkrai, C. Shen, and J. Zhang. fast pedestrian detection using a cascade of boosted covariance features. *IEEE TCSVT*, 18:11401151, 2008. 1

[13] Y. Pang, Y. Yuan, and X. Li. Gabor-based region covariance matrices for face recognition. *IEEE TCSVT*, 18(7):989993, 2008. 1

[14] R. Sivalingam, D. Boley, V. Morellas, and N. Papanikolopoulos. Tensor sparse coding for region covariances. In *ECCV*, page 722735. 1

[15] S. Sra. Positive definite matrices and the s-divergence. *arXiv preprint*, arXiv:1110.1773, 2011. 1

[16] O. Tuzel, F. Porikli, and P. Meer. Human detection via classification on riemannian manifolds. In *CVPR*, 2007. 1

[17] O. Tuzel, F. Porikli, and P. Meer. Region covariance: A fast descriptor for detection and classification. In *ECCV*, page 2006, 589–600. 7

[18] R. Vemulapalli, F. Arrate, and R. Chellappa. Human action recognition by representing 3d skeletons as points in a lie group. In *CVPR*, pages 588–595, 2014. 7

[19] R. Vemulapalli and D. Jacobs. Riemannian metric learning for symmetric positive definite matrices. *arXiv:1501.02393*. 1, 2, 5

[20] R. Vemulapalli, J. Pillai, and R. Chellappa. Kernel learning for extrinsic classification of manifold features. In *CVPR*, pages 1782–1789, 2013. 1

[21] C. Wang, Y. Wang, and A. Yuille. An approach to pose-based action recognition. In *CVPR*, pages 915–922, 2013. 7

[22] L. Wang, J. Zhang, L. Zhou, C. Tang, and W. Li. Beyond covariance: Feature representation with nonlinear kernel matrices. In *ICCV*, page 2015. 1, 7

[23] R. Wang, H. Guo, L. Davis, and Q. Dai. Covariance discriminative learning: A natural and efficient approach to image set classification. In *CVPR*, pages 2496–2503, 2012. 1, 8

[24] W. Wang, R. Wang, Z. Huang, S. Shan, and X. Chen. Discriminant analysis on riemannian manifold of gaussian distributions for face recognition with image sets. In *CVPR*, 2015. 1

[25] C. Wee, P. Yap, D. Zhang, L. Wang, and D. Shen. Constrained sparse functional connectivity networks for mci classification. In *MICCAI*, pages 212–219, 2012. 1

[26] K. Weinberger and L. Saul. Distance metric learning for large margin nearest neighbor classification. *The Journal of Machine Learning Research*, 10:207–244, 2009. 2, 4

[27] E. Xing, A. Ng, M. Jordan, and S. Russell. Distance metric learning with application to clustering with side-information. In *NIPS*, 2009. 2

[28] W. C. Y. Zhu and G. Guo. Fusing spatiotemporal features and joints for 3d action recognition. In *CVPRW*, page 2013, 486–491. 7

[29] L. Yang. Distance metric learning: A comprehensive survey, 2006. 1

[30] X. Yang and Y. Tian. Super normal vector for activity recognition using depth sequences. In *CVPR*, pages 804–811, 2014. 7

[31] C. Yuan, W. Hu, X. Li, S. J. Maybank, and G. Luo. Human action recognition under log-euclidean riemannian metric. In *ACCV*, page 2009, 343353. 1

[32] M. Zanfir, M. Leordeanu, and C. Sminchisescu. The moving pose: An efficient 3d kinematics descriptor for low-latency action recognition and detection. In *ICCV*, pages 2752–2759, 2013. 7